מגישים: רות גוטקוביץ ת.ז 209287085
ניקיטה ליסוקון ת.ז 332684190

תרגיל בית 2 מבוא לבינה מלאכותית

Ex2 Introduction to AI

Gobblet Gobblers

(enhanced TicTacToe)

הקדמה ואדמיניסטרציה

הנחיות כלליות

- תאריך הגשת התרגיל: **29.12.2022**
- את המטלה יש להגיש בזוגות בלבד – בקשות להגשה ביחידים באישור המתרגל האחראי בלבד (ספיר טובול).
- יש להגיש את המטלה מוקלדת בלבד – פתרון בכתב יד לא ייבדק.
- התשובות צריכות להיות כתובות בשפה העברית או באנגלית.
- אפשר לשלוח שאלות בנוגע לתרגיל דרך הפיאצה.
- המתרגלת האחראית על התרגיל: אופק גוטליב.
- בקשות דחיה מוצדקות יש לשלוח למתרגל האחראי בלבד.
- במהלך התרגיל ייתכן שיעלו עדכונים – הודעה תפורסם בהתאם במקרה זה.
- העדכונים מחייבים וזוהי אחריותכם להתעדכן לגביהם עד מועד הגשת התרגיל. עדכונים יופיעו בטופס בצבע צהוב.
- העתקות תטופלנה בחומרה.
- ציון המטלה כולל חלק יבש וחלק רטוב. בחלק היבש נבדוק שתשובתכם נכונה, מלאה, קריאה ומסודרת. בחלק הרטוב הקוד שלכם ייבדק בצורה מקיפה באמצעות בדיקות אוטומטיות – אשר ישוו את המימוש שלכם למימוש שלנו. חשוב לעקוב בזהירות אחר הוראות התרגיל מאחר שסטיות מהמימוש המבוקש עלולות להוביל לכשל בטסט האוטומטי (גם אם המימוש "נכון ברובו"). במהלך הבדיקה האוטומטית יינתן זמן סביר לכל הרצה – כך שכל עוד תעקבו אחר ההוראות, המימוש שלכם יעמוד בהגבלות הזמנים ויחזיר תוצאות טובות מספיק בשביל הטסט.
- מומלץ להסתכל בקוד בעצמכם. שאלות בסיסיות על פייתון שלא נוגעות לתרגיל כדאי לבדוק באינטרנט לפני שאתם שואלים בפיאצה. מומלץ לקרוא את הקוד הנתון על מנת להבין את אופן פעולתו – במקרה שישנם דברים לא מובנים . (לשם כך יש הערות רבות ואף הסבר מורחב על הסביבה!)
- מומלץ לא לדחות את התרגיל לרגע האחרון מאחר שהמימוש וכתיבת הדו"ח עלולים לקחת יותר זמן מהצפוי.
- התייחסות בלשון זכר, נקבה או רבים מתייחסים כלפי כלל המינים.
- ציון התרגיל המקסימלי הינו **110**כיוון שיש בונוס - פירוט על הבונוס בסוף החלק הרטוב

הוראות הגשה

בתוך קובץ זיפ עם השם : HW2_AI_id1_id2.zip

את הדו"ח היבש בפורמט הבא : id1_id2.pdf

ואת הקובץ: subbmision.py שבו אתם ממשים את האלגוריתמים

היכרות עם המשחק

Introduction with the game

לוח וכלי משחק:

לוח משחק של 3X3

2 שחקנים

לכל שחקן: 6 כלים

2 בגודל קטן

2 בגודל בינוני

2 בגודל גדול

המשחק בכללי:

גרסה משופרת של איקס עיגול, השחקן המנצח הוא זה שהצליח למלא שורה, טור או אלכסון בסימנים שלו (הגודל לא משנה).

שחקנים יכולים לזלול שחקנים אחרים ובכך לשנות את הצבע של משבצת קיימת.

להלן סרטון אשר עוזר להבין בכלליות את מהות המשחק:

How to Play Gobblet Gobblers

המשחק בספציפי:

- כל שחקן בתורו מניח גובלין על הלוח, או על משבצת ריקה או על גובלין קטן יותר
- במקום להניח גובלין חדש על הלוח יכול שחקן בתורו לבחור גובלין שלו שנמצא על הלוח ולהזיז אותו למשבצת חוקית
- גובלין יכול לזלול כל גובלין שקטן ממנו (גם אם הם בצבעים זהים)
- ניתן להניח 3 גובלינים אחד על השני (גדול על בינוני כאשר הבינוני על קטן)
- דגש חשוב: במשחק המקורי אין הגבלת צעדים ובגרסה שלנו אנו מגבילים את מספר התורות במשחק. (אם אין נצחון ונעבור את הגבלת הצעדים נחשיב זאת בתור תיקו)
- דגש חשוב: במשחק המקורי חשוב לזכור מה שיש מתחת לכל גובלין שעל הלוח, אצלנו יודעים תמיד (למחשב יש זכרון טוב 🙂)

להלן ספר החוקים הרישמי של המשחק: Gobblet gobblers rules

מה קיבלתם?

קיבלתם 4 קבצים מרכזיים:

1. Gobblet_Gobblers_Env.py סביבת המשחק אותו ישחקו הסוכנים שלכם - הסבר מפורט בהמשך
2. submission.py קובץ שבו תממשו את הסוכנים שלכם וזהו גם הקובץ אותו אתם מגישים
3. game.py מכיל שתי פונקציות שמטרתן להריץ את המשחק ויעזרו לכם לבדיקה עצמית - הסבר מפורט עליהן בהמשך
4. main.py משמש גם כן לבדיקה עצמית

היכרות עם הסביבה

הסביבה איתה תעבדו ממומשת בקובץ Gobblet_Gobblers_Env.py היא מבוססת על סביבה של gym כמו הסביבה איתה עבדתם בתרגיל בית 1.

היא מכילה את המתודות הבאות שקשורות לתפקוד הסביבה:

- **init()** - הקונסטרקטור של הסביבה (המחלקה). ניתן להשתמש בו באופן הבא:

```
env = GridWorldEnv()
```

- **reset()** - מאפסת את לוח המשחק. שמה את כל הכלים בצדדים ולוח המשחק ריק. משתמשים בה באופן הבא `state = env.reset()` הפונקציה מחזירה מצב (כדי להבין מה המשמעות של להחזיר מצב תקראו את הפירוט על הפונקציה get_state()), להלן הלוח במצב מאותחל:

```
        +-------+-------+-------+
B1   B2 |       |       |       |  B1   B2
M1   M2 +-------+-------+-------+  M1   M2
S1   S2 |       |       |       |  S1   S2
        +-------+-------+-------+
        |       |       |       |
        +-------+-------+-------+
```

*האותיות מייצגות את גודל השחקן (B - גדול, M - בינוני, S - קטן)

- **step()** - פונקציה שמקבלת action ומבצעת את הפעולה. הפעולה צריכה להיות tuple בפורמט (pawn,location) כאשר pawn מייצג את השחקן שתרצו להזיז מבין האופציות {2B1,B2,M1,M2,S1,S} ו-location את המקום על הלוח בו תרצו להניח את הכלי. location הוא מספר בין 0 ל - 8 שמייצג את המשבצת על הלוח באופן הבא:

```
+-----+-----+-----+
| 0   | 1   | 2   |
+-----+-----+-----+
| 3   | 4   | 5   |
+-----+-----+-----+
| 6   | 7   | 8   |
+-----+-----+-----+
```

דוגמא לשימוש בפונקציה

```
env.step(("B1", 7))
```

המשמעות: נזיז את גובלין 1B לאריח מספר 7.

- **render()** - מדפיסה את לוח המשחק דוגמא לשימוש בפונקציה

```
env.render()
```

  - שימו לב, כלים שנמצאים על כלים אחרים יסתירו את הכלים שמתחתיהם, כדי לדעת בדיוק איפה כל כלי ראו get_state().
  - שימו לב שיפתח לכם חלון pygame עם גרפיקה נוחה של המשחק ובו תוכלו לראות את המצב של הלוח לאחר שתבצעו (ליתר דיוק הסוכן שלכם יבצע 🙂) צעדים, בנוסף יש הדפסות לקונסולה שתוכלו להיעזר בהן בדיבוג.

* ישנן פונקציות פנימיות רבות נוספות, מוזמנים לקרוא את התיאור שלהם חלקן אפילו יכולות לעזור לכם בכתיבת היוריסטיקה בהמשך, אך אין צורך להתעמק בהן.

בנוסף הסביבה מכילה את הפונקציות הבאות שקשורות לאינטגרציה שלה עם הסוכנים שאתם הולכים לממש בתרגיל זה:

get_state() - פונקציה שמחזירה את המצב הנוכחי של הסביבה, משתמשים בה באופן הבא:

```
env.get_state()
```

והצורה שבה המצב של הסביבה מוחזר הוא במבנה של State (שמוגדר בקובץ Gobblet_Gobblers_Env.py).

get_neighbors() - פונקציה שמקבלת טיפוס מסוג state ומחזירה רשימה שמכילה את כל השכנים שלו (כל המצבים שיווצרו מכל הפעולות החוקיות שאפשר להפעיל על אותו מצב) ברשימה בעצם כל איבר הוא tuple של (action, state) כך תוכלו לעבור על המצבים ולהחזיר בקלות את הפעולה עבור המצב שתבחרו מרשימת השכנים.

is_final() - מבלבל אז חשוב לשים לב! פונקציה שמקבלת מצב מסוים ומחזירה :

None– עבור מצב לא סופי.

0 - אם יש תיקו, אם שני השחקנים ניצחו באותו צעד (מצב תקין) או אם עברו 100 תורות (50 לכל שחקן) ואין הכרעה.

1 - ניצחון של השחקן הראשון. שימו לב! במהלך המשחק, התור של השחקן הראשון מצוין בתור 0 ולא 1!

2 - ניצחון של השחקן השני.

- **play_game()** - פונקציה שמריצה משחק בודד. הפונקציה הנ"ל מקבלת מחרוזות שמתארות כל סוכן כפי שמתואר בתצלום של המילון בפונקציה הבאה. בנוסף יש

דוגמא בהמשך. הפונקציה מחזירה מיהו המנצח ומדפיסה זאת למסך. מחזירה תוצאות כמו שמחזירה is_final().

- **play_tournament()** - פונקציה המריצה מספר play_games לפי ערך num_games שתתנו לה (אנו נבדוק את הקוד שלכם עם num_games = 50) ובכך מחזירה באחוזים כמה ניצחונות יש לכל שחקן וכמה תיקו היו במשחק. הפונקציה הנ"ל מקבל מחרוזות שמתארות כל סוכן כפי שמתואר בתצלום של המילון הנ"ל. דוגמא בהמשך.

```
"human": submission.human_agent,
"random": submission.random_agent,
"greedy": submission.greedy,
"greedy_improved": submission.greedy_improved,
"minimax": submission.rb_heuristic_min_max,
"alpha_beta": submission.alpha_beta,
"expectimax": submission.expectimax
```

num_games * הוא לא מספר המשחקים שרצים בפועל, בפועל רצים פי 2 משחקים. הוא מייצג כמה משחקים יש בהם כל שחקן הינו השחקן הראשון. (תורו לשחק ראשון)

מתחילים לכתוב!

חלק א - היכרות עם הסביבה (4 נק')

1. (יבש: 1 נק') כנסו לקובץ main והריצו את השורה שמסומנת בהערה ומעליה כתוב 1#PART השורה מריצה את המשחק עם שני סוכנים אנושיים, שחקו אחד נגד השני עד לניצחון וצרפו את ההדפסה של המצב הסופי.

```
            +------+------+------+
        B2  |  M1  |      |      |       B2
        M2  +------+------+------+  M1   M2
        S2  |      |  S1  |      |  S1   S2
            +------+------+------+
            |      |  B1  |  B1  |
            +------+------+------+
time for step was 10.869222402572632
winner is: 1

Process finished with exit code 0
```

2. (יבש: 2 נק') האם ניתן לגרום בתור של סוכן מסוים לביצוע פעולה שתגרום לסוכן האחר לנצח? אם כן הסבירו מדוע מצב שכזה יכול לקרות וצרפו את שני הצעדים האחרונים במשחק שכזה, אם לא, הסבירו מדוע לא יתכן מצב כזה?

שחקן צהוב יכול להזיז M1 שכבר נמצא על הלוח למיקום אחר בלוח ולשחרר גובלין של יריב שנמצא מתחתיו (S1) ובכך שחקן הכחול ינצח.

```
                +------+------+------+
   B1    B2     |  M1  |  S1  |      |        B2
                +------+------+------+        M2
                |  M1  |  S2  |      |        S2
                +------+------+------+
                |  B1  |      |  M2  |
                +------+------+------+
time for step was 5.38032865524292
player 1
insert action
insert pawn: M1
insert location: 7
                +------+------+------+
   B1    B2     |  S1  |  S1  |      |        B2
                +------+------+------+        M2
                |  M1  |  S2  |      |        S2
                +------+------+------+
                |  B1  |  M1  |  M2  |
                +------+------+------+
time for step was 13.92495059967041
winner is: 1

Process finished with exit code 0
```

3. (יבש: 1 נק') מהו המספר המקסימלי של אפשרויות לפעולות שונות (בהנחה שכל פעולה חוקית) שניתן לעשות בתור?

לכל גובלין יש לכל היותר 9 מיקומים על הלוח עליהם ניתן לשים אותו. סה"כ יש 6 גובלינים לכל שחקן, לכן יש לכל היותר 6*9=54 אפשרויות.

חלק ב - Improved Greedy

(15 נק')

בקובץ submission.py שקיבלתם ממומש עבורכם סוכן greedy שמשתמש בהיוריסטיקה לא חכמה שנקראת dumb_heuristic2 היא מחשבת את כמות השחקנים הגלויים (לא נמצאים מתחת לשחקן אחר) שיש לסוכן שלה על הלוח.

*שימו לב! ממומשת ב-submission היוריסטיקה שנקראת dumb_heuristic1 איננו משתמשים בה בשום מקום אך מה שהיא מבצעת הוא : מחזירה 0 - אם המצב אינו מצב סופי. מחזירה 1 אם ניצחנו ו-1- אם הפסדנו או שהיה תיקו. אתם מוזמנים להסתכל עלייה בכדי להבין כיצד להשתמש ב - is_final().

1. (יבש: 6 נק') הגדירו היוריסטיקה משלכם להערכת מצבי המשחק, כתבו נוסחה מפורשת עבור היוריסטיקה. מוזמנים להוסיף תרשים או פירוט מפורט של מה היוריסטיקה עושה בהניתן מצב הלוח והשחקן שמשחק. בחרו בהסבר שמות ברורים ומשמעותיים.

יוריסטיקה החדשה תיתן יותר משקל לרצפי גובלינים גדולים יותר. אם יש רצף של גובלין אחד בשורה או עמודה יוריסטיקה תיתן 1 ,אם יש רצף של 2 גובלינים בשורה, עמודה או אלכסון אז יוריסטיקה תיתן עליהם 10,אם יש רצף של 3 גובלינים בשורה, עמודה או אלכסון יוריסטיקה תיתן עליהם 100.
כלומר אנו נותנים יותר "משקל" לצעדים שיקרבו אותנו לנצחון, שזה 3 גובלינים ברצף שלא מכוסים ע"י גובלינים של היריב

2. (רטוב: 5 נק') ממשו בקובץ submission.py את `greedy_improved` וממשו את היוריסטיקה החכמה שלכם תחת הפונקציה (`smart_heuristic.` אל תשנו חתימות של פונקציות זה יכשיל את הטסטים!)
greedy_improved הינה פונקציה המקבלת:
   - curr_state - מצב נוכחי.
   - agent_id - השחקן שמשחק כעת - אם זהו השחקן הראשון יועבר 0 אם זהו השחקן השני יועבר 1.
   - time_limit - הגבלת הזמן (בשלב זה אתם יכולים להתעלם ממנה, היא תיהיה רלוונטית באלגוריתמים הבאים).
   הפונקציה מחזירה את הפעולה שהסוכן צריך בוחר לבצע כזוג סדור (pawn, location) כפי שפורט בהסבר על הסביבה (זהה לקלט של מתודת step()).
3. (יבש : 2 נק') הסבירו את המוטיבציה לשינויים שביצעתם בהיוריסטיקה האם לפי דעתכם סוכן חמדן המבוסס על היוריסטקיה שלכן (greedy_improved) ינצח את הסוכן החמדן (greedy)? אם כן, פרטו מדוע.

בנינו יוריסטיקה שנותנת משקל נוסף אם פעולה יוצרת זוגות או שלשות בלוח (בשורה, עמודה או אלכסון) אצל סוכן שלנו. יוריסטיקה נותנת משקל 1 לכל באחדות, משקל 10 לכל הזוגות ומשקל 100 לכל השלשות כאלה ערך סופי שיוריסטיקה תחזיר זהו סכום משקלים של אחדות, זוגות ושלשות שספרנו.
יוריסטיקה הזו עדיפה על זאתי שקיבלנו כי היא נותנת עדיפות לפעולות שיכולות ליצור זוגות ושלשות, כאשר תנאי לניצחון במשחק הוא יצירת שלשה בלוח כפי שתיארנו מקודם.

4. (רטוב: 2 נק') הריצו את השורות שנמצאות בהערה ומעליהן 2#PART וצרפו את התוצאות שיודפו למסך כתוצאה מכך. אתם בעצם תריצו:
   - חמדן נגד אקראי (random)

```
time for step was 0.00500035285949707
winner is: 2
ties:  2.0 %  greedy player1 wins:  92.0 %  random player2 wins:  6.0
```

   - חמדן משופר נגד אקראי (random)

```
time for step was 0.6121830940246582
winner is: 2
ties:  3.0 %  greedy_improved player1 wins:  88.0 %  random player2 wins:  9.0
```

   - חמדן נגד חמדן משופר

```
time for step was 0.5498743057250977
winner is: 1
ties:  0.0 %  greedy player1 wins:  0.0 %  greedy_improved player2 wins:  100.0
```

חלק ג - RB heuristic MiniMax

(20 נק')

1. (יבש: 2 נק') מה היתרונות והחסרונות של שימוש בהיוריסטיקה קלה לחישוב לעומת היוריסטיקה קשה לחישוב בהינתן שהיוריסטיקה הקשה לחישוב יותר מיודעת מהקלה לחישוב (נותנת אינפורמציה טובה יותר לגבי השאלה מהו מצב טוב)? בהינתן שאנו ב-min max מוגבל משאבים.

יתרונות: יוריסטיקה קלה לחישוב לוקחת פחות זמן חישוב, פחות פעולות דרושות לביצוע ופחות מידע צריך לניתוח.

* אלגוריתם מוגבל משאבים כוונה בהגבלה על עומק חיפוש, גודל יוריסטיקה לא ממש משחק תפקיד בכך

חסרונות: יוריסטיקה הקלה לחישוב שפחות מיודעת לא בהכרח תוביל אותנו למטרה במסלול אופטימלי, אם בכלל. זאת מכיוון שאנחנו מוגבלים בידע על העולם ויכולים לקבל החלטה שגויה וללכת לכיוון הלא נכון במשחק.

2. (יבש: 3 נק') דני מימש את האלגוריתם RB-heuristic-MiniMax והריץ אותו ממצב s בו קיימת פעולה בודדת שמביאה לניצחון. דני נדהם לגלות שהאלגוריתם לא בחר בפעולה זו. האם בהכרח יש טעות באלגוריתם שדני כתב? אם כן, הסבר מה הטעות, אחרת, הסבר מדוע האלגוריתם פעל באופן זה.

מצב בו השחקן יכול לנצח בתור הבא אך האלגוריתם minimax לא מבצע מהלך זה, מסמן לנו כי האלגוריתם מצא מספר דרכים להגיע לנצחון. כאשר כל דרך היא כמובן שונה מדרך אחרת, ולא בהכרח שהנצחון יתקבל בתור הבא.
אם האלגוריתם minimax מצא רק דרך אחת להגיע לנצחון (נצחון בתור הבא שלנו), האלגוריתם היה מחזיר לנו את המהלך שיוביל אותנו לנצחון בתור הבא, נצחון הוא למעשה מצב היעד שלנו ולכן יבחר כי הוא המקסימלי (הטוב ביותר עבורנו).
אחרת יש מספר אפשרויות למהלכים שיובילו אותנו לנצחון (לא בהכרח בתור הבא שלנו) במצב זה minimax יחזיר את אחד המהלכים שיובילו בסופו של דבר לנצחון, אך לא הוגדר איך נבחר צעד במקרה של שוויון בין מספר מהלכים (הם שווים כי כולם יובילו לנצחון).

3. (יבש: 3 נק') למדתם בהרצאות ובתרגולים גישה שנקראת anytime search. כיצד היא מתמודדת עם הגבלת הזמן ? איזה בעייה נפוצה יש באיטרציה האחרונה ואיך פותרים אותה?

מריצים אלגוריתם לעומק שונה, כל פעם מגבירים עומק עד שיגמר הזמן. מחזירים תוצאת חישוב מהרצה אחת לפני האחרונה
בעיה נפוצה היא בכך שאיטרציה אחרונה בכלל לא מועילה לנו, אבל היא גם הכי כבדה לחישוב מבחינת כמות הצמתים שמפתחים.
ניתן לפתור אותה בכך שנסדר ילדים לפי יוריסטיקה מסוימת ובהרצה אחרונה נפתח רק את הילדים עם מדדי יוריסטיקה גבוהים ביותר ולשאר הילדים נשתמש בפונקציית מחיר.

4. (יבש: 4 נק') נניח שבסביבה היו K שחקנים במקום 2 (תחשבו על משחק כללי לוא דווקא המשחק שלנו, אך עדיין משחק סכום אפס). אילו שינויים יהיה צריך לעשות במימוש סוכן Minimax?
   a. בהינתן שכל סוכן רוצה לנצח ולא אכפת לו רק ממכם
      סוכן אמור לקחת בחשבון רק ששחקן הבא בתור יבחר בערך שמזיק לו, שאר השחקנים מתחרים אחד בשנים ולא מודעים בכלל למצב שלנו.
      לכן אלגוריתם יעבוד בדיוק באותו אופן כמו מקודם, רק ששחקן שישחק מולנו בתור הבא לא יהיה בוודאות אותו שחקן כמו מקודם

   b. בהנחה שכל סוכן שונא אתכם והדבר היחיד שאכפת לו זה שלא תנצחו

      בפיתוח העץ נבחר במסלול המבטיח שנקבל ערך המקסימלי האפשרי בהנחה ששאר השחקנים יבחרו בערך שהכי "יזיק" לנו. כלומר אותו פיתוח כמו בminmax רק שk-1 שחקנים יבחרו בערך min הפעם ורק אנחנו נחפש ערך max.

5. (רטוב: 8 נק') עליכם לממש את הפונקציה `rb_heuristic_min_max` בקובץ submission.py. שימו לב כי הסוכן מוגבל משאבים, כאשר המשתנה time_limit מגביל את מספר השניות שהסוכן יכול לרוץ לפני שיחזיר תשובה.(בסעיף זה אסור להשתמש בגיזום כדי לפתור את הבעייה - אל תדאגו בחלק הבא יהיה לכם גיזום).

* מומלץ להריץ את הסוכנים שלכם אחד נגד השני כמו בחלק א ולראות כי אתם מקבלים תוצאות הגיוניות

חלק ד - **Alpha_Beta**

(20 נק')

1. (רטוב: 10 נק') כעת אתם ממשו סוכן אלפא בטא, בקובץ submission.py ממשו את הפונקציה `alpha_beta` כך שתבצע גיזום כפי שנלמד בהרצאות ובתרגולים.
2. (יבש: 3 נק') האם הסוכן שמימשתם בחלק ד' יתנהג שונה מהסוכן שמימשתם בחלק ג' מבחינת זמן ריצה ובחירת מהלכים?

במקרים רבים סוכן האלפא בטא צפוי לרוץ פחות זמן מאשר הסוכן בחלק ג, מכיוון שהוא לא עובר בחלקים מיותרים בעץ בזכות הגיזום, לכן נוכל להחליט על המהלך המתאים בצורה מהירה יותר. בנוסף, יתכן מקרה שבו ה"בנים" מסודרים בצורה הכי פחות טובה לצורך הגיזום כמו שראינו בהרצאה (אם מחפשים מקסימום, אז כל בן "משפר" את הערך כך שהאחרון הטוב ביותר) ובמקרה זה שניהם עוברים על כל העץ אך האלפא בטא אמור להיות דווקא פחות מהיר, בגלל "עלות" התחזוקה של ערכי אלפא ובטא.
מבחינת בחירת מהלכים - יתכן שהמהלכים שסוכן אלפא בטא יבחר יהיו שונים, כי העובדה שהוא לא עובר על חלק מהענפים שלא רלוונטיים, יכלה לגרום לו להגיע עמוק יותר בעץ תוך אותה מגבלת זמן, ובהתאם לערכים הנוספים שהוא גילה לבחור מהלך אחר.

3. (יבש: 2 נק') למדתם מספר שיטות לשיפורים של alpha_beta אחת מהשיטות הללו נקראת ספריות פתיחה / סיום. מה שיטה זו מבצעת ומדוע היא עוזרת לגזום נתח מהעץ?

השיטה של ספריות פתיחה/סיום הינה שיטה שבה אנו נזכור רצפי משחק/תוכניות פעולה למצבים נפוצים, ונשתמש בהם אם נגיע למצב שהוא חלק מרצף זה. שיטה זו עוזרת לגזום חלק מהעץ כי כעת אין צורך לשמור את כל העץ במלואו אלא כאשר נגיע למצב נפוץ כלשהו נשתמש בתוכנית פעולה שכבר שמורה בזיכרון ולא נצטרך למצוא תוכנית פעולה.

4. (יבש: 3 נק') למדתם על גישה נוספת שנקראת טבלאות מצבים. מה השיטה מבצעת, איך היא שומרת על נכונות, ומדוע היא עוזרת "לגזום" מהעץ?

בשיטה זו אנו שומרים את ערכי המינימקס בטבלה עבור המצבים השונים, כאשר לכל מצב נשמור רשומה השומרת לכל מצב את ערך המינימקס ואת עומק החיפוש שהערך משקף. השיטה עוזרת לגזום את העץ כי כאשר בעתיד נגיע לאותו מצב נוכל להשתמש בערך המינימקס השמור לאותו מצב במקום לחפש מחדש ערך זה. השיטה שומרת על נכונות כי אנו ניקח ערך מהטבלה רק עבור מצב שבו עומק בחיפוש שהערך משקף גדול שווה מהעומק שנותר בקריאה הנוכחית. אחרת אם לא נשתמש בערך זה אזי האסטרטגיה בעומק כלשהו d תיקבע על ידי ערכי מינמקס מעומקים קטנים מ-d. בצורה זאת אנו שומרים כי משפט ההבטחה של מימקס מתקיים ולכן השיטה שומרת על נכונות.

5. (יבש: 2 נק') בהרצה של $\alpha\beta - Minimax$ המתכנת מחק בטעות את הצעד המומלץ ונשאר רק עם ערך המינימקס של העץ. כיצד ניתן לשפר את יעילות ההרצה החוזרת באמצעות שינויים פשוטים באלגוריתם? תארו את התנהגות האלגוריתם המתוקן. הסבירו כיצד יתנהג במקרה הטוב ביותר, במקרה הרע ביותר, ובמקרה הכללי.

נניח כי קיים עבור השורש אנו יודעים את הערך המינמקס אשר היה מתקבל מהרצת האלגוריתם, אבל לא ידוע מאיזה בן הוא הגיע ולכן לא ידוע איזה פעולה יש לעשות. אזי נבצע את השינוי הבא- מכיוון ואנו יודעים את הערך של השורש, אנו יודעים גם את הערך של אלפא, אזי בפונקציה max_value נוכל להגדיר את v להיות הערך של אלפא, ובמהלך הריצה כאשר נמצא בן של השורש אשר ערכו שווה לערך של אלפא נחזיר אותו.
(התייחסתי בתשובה רק לבחירת פעולה בודדת ולא למשחק כולו.)
במקרה הטוב ביותר, הבן הראשון שנבדוק אותו ערכו יהיה שווה לערך האלפא ואז נחזירו ולא יהיה צורך לעבור על כל שאר הבנים של השורש. כלומר ניתן יהיה לחסוך משמעותית בזמן ובמקום.
במקרה הרע ביותר הבן אשר ערכו שווה לערך האלפא יהיה הבן האחרון אותו נבדוק ואז השיפור שהכנסתי לא יעזור לשפר את היעילות. ובעצם נצטרך להריץ מחדש את כלל האלגוריתם.
במקרה הכללי השינוי עשוי לשפר כי ברגע שנמצא את הבן המתאים נפסיק לעבור על שאר הבנים וככה נוכל לחסוך בזמן ובמקום עי לא מפתחים את כלל הבנים.

חלק ה - **Expectimax**

(20 נק')

1. (יבש: 2 נק') סוכן Minimax (עם או בלי שיפורים) מניח כי היריב בוחר בכל צעד בפעולה האופטימלית עבורו, מה בעייתי בגישה הזו ואיך אלגוריתם Expectimax מתגבר על הבעייתיות שתיארתם?

הבעייתיות בגישה הנל כי לא תמיד היריב בוחר את הפעולה האופטימלית עבורו ואילו אנו מסתמכים על עובדה זאת על מנת לבחור את המהלך שאותו נבצע.
האלגוריתם מתגבר על בעייתיות זאת בצורה הבאה- אנו מחשבים את ערכי המקס

ללא שינוי ואילו ערכי המינימום ישתנו- כעת, אנו נחשב את המקרה הממוצע (על ידי שימוש בהסתברות לקבלת מצב כלשהו) ובצורה זאת אנו לא מניחים כי היריב בוחר בפעולה האופטימלית עבורו אלא מניחים כי הוא בוחר בפעולה הממוצעת.

2. (יבש: 3 נק') בהנחה ואתם משתמשים באלגרותים Expectimax נגד סוכן שמשחק באופן רנדומלי לחלוטין באיזה הסתברות תשתמשו? ומדעו?

במידה ומדובר ביריב המשחק באופן רנדומלי לחלוטין אנו נבחר להשתמש בהסתברות אקראית לקבלת מצב כלשהו כי אין יוריסטיקה/הסתברות מסוימת שקובעת לאילו מהלכים יש עדיפות.

3. (יבש: 5 נק') עבור משחקים הסתברותיים כמו שש בש, בהם יש מגבלת משאבים, משתמשים באלגוריתם .RB-Expectimax הניחו כי ידוע שהפונקציה היוריסטית h באלגוריתם Expectimax-RB מקיימת $\forall s: -6 \leq h(s) \leq 6$

a. איך ניתן לבצע גיזום לאלגוריתם זה? תארו בצורה מפורטת את תנאי הגזימה, והסבירו את הרעיון מאחוריו.

לפי הנתון בשאלה בתחילת הריצה נגדיר את בערכים אלפא ובטא להיות: $\alpha = -6,\ \beta = 6$. הרעיון הוא שאין צורך להגדיר את אלפא ובטא להיות $\alpha = -\infty,\ \beta = \infty$ כי ידוע שהערכים שהיוריסטיקה מחזירה הם בתחום $-6 \leq h(s) \leq 6$ כלומר לא ייתכן כי נקבל ערכים מתחום אחר ולכן נגדיר את אלפא ובטא בצורה המתאימה למשחק הנתון.

b. הציגו דוגמה להיוריסטיקה כזאת עבור המשחק בתרגיל שלנו וצרפו דוגמא ללוח עבור כל אחד מהמצבים הבאים:
   - מצב המקיים $h(s) = 6$
   - מצב המקיים $h(s) = -6$

דוגמה ליוריסטיקה המקיימת $-6 \leq h(s) \leq 6$:

$$h(s) = \begin{cases} 6, & if\ state\ isnt\ final \\ -6, & if\ state\ is\ final \end{cases}$$

דוגמא ללוח המקיים $h(s) = 6$:
ניתן לראות כי המצב אינו מצב סופי
ולכן נקבל $h(s) = 6$ לפי הגדרת
היוריסטיקה.

```
player 0
          +------+------+------+
B1   B2   |      |      |      |  B1   B2
M1   M2   +------+------+------+  M1   M2
S1        |      |      |      |  S1   S2
          +------+------+------+
          |      |      |  S2  |
          +------+------+------+
```

דוגמא ללוח המקיים
$h(s) = -6$: ניתן לראות
כי מדובר במצב סופי
(השחקן הצהוב ניצח ולכן
נקבל $h(s) = -6$ לפי
הגדרת היוריסטיקה.

```
player 0
            +------+------+------+
B1          |  B2  |      |      |        B2
M1    M2    +------+------+------+  M1
S1          |      |      |      |  S1
            +------+------+------+
            |  B1  |  S2  |  M2  |
            +------+------+------+
```

4. (רטוב: 10 נק') כעת תממשו סוכן expectimax ע"י כך שתממשו בקובץ submission.py את הפונקציה `expectimax,` תחת ההנחה כי היריב שלנו אוהב אקשן ומחליט שאת כל הפעולות מבצע בהסתברות שווה למעט שתי סוגי פעולות:
   a. לפעולות שבהן יכול לאכול חייל אחר יש הסתברות גבוהה פי 2 משאר הפעולות
   b. הוא מעדיף חיילים קטנים (בגודל S) ולכן לפעולה שמערבת שחקן S יש גם כן הסתברות גבוהה פי 2 משאר הפעולות

בונוס בונוס בונוס בונוס בונוס

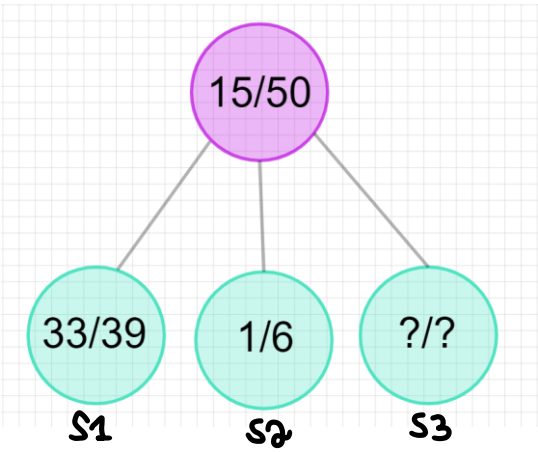
כפי שאתם יודעים האלגוריתמים שמימשתם הינם אדברסיאלים, משמע מתחרים אחד בשני, ולכן אנו מזמינים אתכם לכתוב סוכן שיתחרה בסוכנים של שאר הסטודנטים בקורס עליכם לממש אותו תחת הפונקציה supre_agent יש לכם יד חופשית בבחירת האופן בה תממשו את הסוכן (יכולים לבחור אפילו אחד מהאלו שמימשתם בתרגיל). שני הזוגות שינצח מול הכי הרבה קבוצות אחרות יקבלו 10 נק' בונוס לציון הסופי של תרגיל הבית הנ"ל. ארבעת הזוגות שאחריהם יקבלו 5 נק' בונוס.

שאלה פתוחה - **Monte Carlo Tree Search**

(21 נק')

1. (2 נק') הסבירו את המונחים הבאים:
   a. Exploitation – focusing on moves that promise us some minimum achivement
   b. Exploration – focusing on moves where uncertainty about evaluation is high

אנדריי ואופק החליטו לשחק קאטאן (נניח שזה משחק לשני אנשים), הם החליטו שהמשחק פשוט מידי והחליטו להוסיף את ההרחבות: "ערים ואבירים" ו"יורדי הים". אחרי שהבינו שהגזימו וכעת אינם בטוחים מה הצעד החכם ביותר ובגלל שה-branching factor גדול מאוד החליטו להשתמש ב- MCTS (Monte Carlo Tree Search) (דגש - זהו משחק סכום אפס שנגמר בניצחון של אחד הצדדים)



להלן עץ שמתאר שני שלבים במשחק. כחול זה תור של אופק וורוד זה תור של אנדריי (פחות רלוונטי לשני הסעיפים הראשונים)

2. (2 נק') עבור העץ הנ"ל שמתאר שני שלבים במשחק עבור העלה עם ה- (?) השלימו מהו הערכים שאמורים להיות במקום הסימני שאלה.

1/5

סה"כ יש 50 דגימות לכן 50=39+6+5

שחקן כחול משחק נגד שחקן הוורוד לכן

50-33-1-1=15

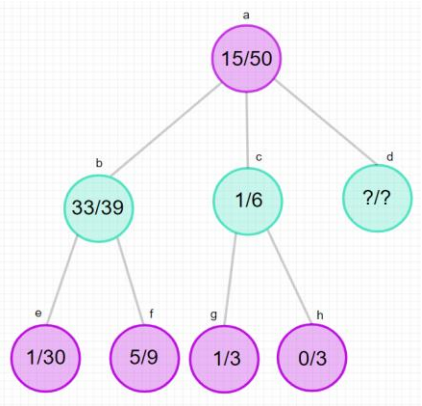
3. (7 נק') חשבו את $UCB_1(s)$ עבור הצמתים הכחולים עם ערך $C = \sqrt{2}$, השוו בין הערכים שיצאו לכם בחישוב ה - $UCB_1(s)$, הסבירו מה המשמעות שלהם. האם יש

צמתים (רק מהכחולים) שהערך שלהם שונה מהותית אחד מהשני אך ה - $UCB_1(s)$ דומה? אם כן הסבירו מדוע מצב כזה קורה ע"פ העקרונות של UCB_1

$$UCB_1(s1) = \frac{33}{39} + \sqrt{2} * \sqrt{\frac{\ln(50)}{39}} = 1.29$$

$$UCB_1(s2) = \frac{1}{6} + \sqrt{2} * \sqrt{\frac{\ln(50)}{6}} = 1.31$$

$$UCB_1(s3) = \frac{1}{5} + \sqrt{2} * \sqrt{\frac{\ln(50)}{5}} = 1.45$$



שני הצמתים s1, s2 בעלי ערכים שונים אך יש להם ערכי UCB קרובים מאוד. לפי הנלמד בכיתה מצב זה מתרחש מכיוון ש-UCB רוצה לאזן, כלומר מצד אחד לנסות ולבדוק צמתים שלא נוסו מספיק פעמים (s2) ומצד שני לנסות ולבדוק צמתים מבטיחים יותר (s1) ולכן מתקבלים ערכים קרובים מאוד.

כעת חושפים לנו המשך של שני השלבים שראינו והעץ כעת נראה כך:

4. (4 נק') מצאו וציינו מיהו הצומת הבא שיפותח.

בהנחה ובחרנו עבור הצמתים הירוקים את הצומת בעל ה-UCB_1 המקסימלי אזי הצומת שנבחר הוא צומת d אשר עבורו לא קיימים בנים ולכן אין צומת הבא שיפותח. אחרת, נניח כי אנו מתייחסים אך ורק לצמתים ברמה התחתונה ביותר ועלינו לבחור מבין ארבעת הצמתים הנתונים את הצומת הבא שיפותח.

על מנת למצוא את הצומת הבא שיפותח נחשב את UCB_1 עבור כלל הצמתים:

$$UCB_1(e) = \frac{1}{30} + \sqrt{2} * \sqrt{\frac{\ln(39)}{30}} = 0.53$$

$$UCB_1(f) = \frac{5}{9} + \sqrt{2} * \sqrt{\frac{\ln(39)}{9}} = 1.45$$

$$UCB_1(g) = \frac{1}{3} + \sqrt{2} * \sqrt{\frac{\ln(6)}{3}} = 1.42$$

$$UCB_1(h) = \frac{0}{3} + \sqrt{2} * \sqrt{\frac{\ln(6)}{3}} = 1.09$$

הצומת הבא שנבחר בו הינו צומת f (בוחרים את הערך המקסימלי המתקבל).

5. (6 נק') בהנחה ולו יש בן יחיד שלאחר סימולציה ממנו מתקבל הפסד של אופק. ציירו את העץ החדש הנוצר (שנו את ערכי שאר הצמתים בהתאם) לאחר שנפעפע את המידע על הסימולציה שהתרחשה.

בהנחה והצומת שנבחר הוא צומת f והתקבל הפסד של אופק – מתקבלים הערכים הבאים- מסומנים בציור. סימנתי אך ורק ערכים של צמתים אשר השתנו בעקבות ההפסד. (יתר הערכים של יתר הצמתים נותרו ללא שינוי).